基于弹性定额值的分组轮询调度算法

刘桂开 高 蕾
(湖南科技大学计算机科学与工程学院 湘潭 411201)

摘 要 提出了一种新的适用于变长分组的调度算法——弹性定额值轮询调度算法(Resilient Quantum Round Robin,RQRR),与现有算法不同,该算法中每个数据流的定额值不是固定不变的,定额值的生成依赖于前一个轮次中各个数据流的发送情况。理论分析表明,RQRR可以保证数据流之间具有较好的公平性,它的公平性度量具有上界值7Max−1,其中Max为分组的最大长度。RQRR对每个分组的处理复杂度为O(1),易于实现、适用于高速网络。

关键词 分组调度,轮询,弹性定额值,公平性,实现复杂度
中图法分类号 TP393 **文献标识码** A

RQRR:A New Fair and Efficient Packet Scheduling Algorithm

LIU Gui-kai GAO Lei
(School of Computer Science and Engineering,Hunan University of Science and Technology,Xiangtan 411201,China)

Abstract Aiming at variable length packet queues,a new packet scheduling algorithm named Resilient Quantum Round Robin(RQRR)was presented. Different from existent algorithms,the quantum given to each of the flows in a round is not fixed and is calculated depending on the transmission situation of all the flows in the previous round. The theoretical analyses show that the relative fairness measure of RQRR has an upper bound of 7Max−1,where Max is the largest size of the packets. RQRR takes O(1)processing complexity per packet and is simple to implement at high-speed networks.

Keywords Packet scheduling,Round robin,Resilient quantum,Fairness,Implementation complexity

1 引言

分组调度是指按怎样的顺序将到达网络节点的数据分组通过输出链路传送出去,网络的交换节点都要依赖分组调度方法来对到达该节点的所有分组进行缓存、排队和输出。最简单的调度就是传统路由器所采用的先来先服务(First Come First Serve,FCFS)机制,按分组到达的先后顺序对分组进行调度和输出。FCFS虽然实现简单,但无法保证数据流共享网络资源的公平性,如一个突发性强的数据流在短时间内发送大量的数据包,就会影响其他数据流的服务,增加数据流的传送时延。公平性强调的是如果一个数据流所要求的网络资源在它所应该共享的范围之内,那么它的服务就不应该受到网络中其他数据流的影响。好的调度算法应该具有公平、有效和低复杂度等特性。

分组调度算法主要分为两大类:基于时间戳(Timestamp)的算法和基于轮询(Round-robin)的算法。基于时间戳的调度算法是为每个分组维持两个时间戳,标明其开始服务和结束服务的时间,然后对时间排序,选择具有最小结束服务时间的分组发送,如FQ(Fair Queuing)[1]、WFQ(Weighted Fair Queuing)[2]、WF^2Q(Worst-case Fair Weighted Fair Queuing)[3]、VC(Virtual Clock)[4]、SCFQ(Self-Clocked Fair Queuing)[5]等,这类算法通过维持虚拟时钟对最理想的分组调度算法模型GPS(Generalized Processor Sharing)[6]进行近似模拟,具有良好的时延性能和公平性。但是,由于要涉及到对系统中每个分组时间的排序计算,这类算法其时间复杂度至少为O(log$N$)(N为系统中活动业务流的数目),不适合应用于高速网络设备,因此复杂度相对较低的基于轮循策略的算法受到重视。

基于轮询的分组调度是调度器(Scheduler)循环地对每个业务流的队列进行轮流服务来发送队列中的分组,大多具有O(1)的时间复杂度,且实现简单,比较适合在高速网络中使用。但简单的轮询算法RR(Round Robin)和加权轮询算法WRR(Weighted Round Robin)在变长分组环境下都存在不公平性,如设两个流的权重相等,一个流的分组大小为100个字节,另一个流的分组大小为500个字节,这样第二个流得到的服务实际上就是第一个流的5倍。所以,在基于轮询的调度算法中需要考虑分组长度才能保证调度的公平性。DRR(Deficit Round Robin)[7]算法能支持变长分组的调度,且算法的时间复杂度是O(1),因而得到了广泛关注和应用[8-11]。DRR的原理是为每个流分配一个定额值(Quantum),用来衡量每个流在DRR的一次服务中传送的字节数,同时还为每一个流设置一个余额计数器DC(Deficit Counter),用来保存一

到稿日期:2012-10-07 返修日期:2013-03-05 本文受湖南省科技厅科技计划项目(2010GK3045),湖南省教育厅科研项目(10C0687)资助。
刘桂开(1969−),男,博士,高级工程师,主要研究方向为宽带IP技术、无线网络技术,E-mail:guikailiu@gmail.com;高 蕾(1987−),女,硕士生,主要研究方向为计算机网络技术。

个流在 DRR 的一次服务中没有用完的定额值，一个流可以通过 DC 把它上次没有用完的定额值带到下一次服务中。这样在一段较长的时间内，一个流所得到的服务基本上与它所对应的定额值成正比，可以说 DRR 算法是公平的，适合应用在分组长度可变的网络中。但 DRR 也存在一些不足，首先它只能提供长时间尺度的公平性，从短时间尺度来看，会导致数据流的输出有很大的突发性；其次，它的时延特性也不理想，有的队列可能长时间得不到服务；另外，DRR 要求发送分组前必须知道分组的长度，增加了实现的开销。针对 DRR 存在的不足，在许多文献中都提出了对 DRR 的改进，文献[12]提出了具有优先服务机制的 PNDRR 算法，在 DRR 的基础上采用漏桶与虚令牌分配机制相结合的策略，降低业务的调度延迟时间。文献[13]通过在节点处加入基于网络演算的流量整形器，弥补了 DRR 不能以较为平滑方式调度输出的缺陷，提高了网络服务质量。文献[14]针对宽带无线接入中不同业务的 QoS 要求，提出了 Improvised-Modified Deficit Round Robin (I-MDRR)算法，其在满足实时业务 QoS 要求的同时，吞吐量也得到了提升。文献[15]提出了分域调度的思想，即采用承载组内的 SDRR(Smoothed DRR)调度算法和域内最长队列优先调度算法来降低业务调度的相对复杂度和时延。M-DRR[16]考虑到无线通信的特点，结合信道条件和 HARQ 技术应用，对无线分组数据进行调度，以提高 IEEE802.16e 系统的吞吐量并提供一定的 QoS 保证。文献[17]提出的算法称为 DeDuplication-aware Deficit Round Robin(D2DRR)，用于灵活处理航空电子网络中的突发数据流和重复数据，以减轻网络负载和节省网络的带宽。对 DRR 算法的改进，在一定程度上弥补了 DRR 算法的不足，但改进后的算法仍然是以 DRR 为基础，需要给数据流分配固定的定额值(Quantum)。

本文提出了一种完全不同于 DRR 的适用于变长分组的调度算法，其显著特点是每个队列允许发送分组的定额值是动态变化的，而且一个轮次中定额值的计算依赖于前一个轮次中各个数据流的发送情况，所以称之为弹性定额值轮询分组调度算法(Resilient Quantum Round Robin，RQRR)。在 RQRR 算法执行中，一个轮次中获得很少服务的数据流会在下一个轮次中给予更多的服务机会，可以保证数据流之间的调度公平性。与 DRR 相比，RQRR 对定额值的计算都是在分组发送之后进行，所以发送分组前不需要知道分组的长度。

2 RQRR 算法

RQRR 算法包括初始化、入队列排队和出队列调度等 3 个过程。初始化过程是指当网络节点有分组调度的需求时，对分组调度模块进行初始化。每个数据流对应一个队列，入队列排队过程是指一个新的分组到达时，进入到与它所属数据流对应的队列中排队，等候发送。出队列调度过程描述的是怎样从不同数据流对应的队列中选择分组，并将选取的分组通过输出链路发送到网络中。使用 RQRR 算法的情形如图 1 所示，假设有 n(n 为大于 0 的自然数)个数据流共享一条输出链路，每一个数据流的分组在发送到输出链路之前，进入到数据流所对应的队列，等候发送。分组调度模块使用 RQRR 算法来调度分组的传送，使输出链路的带宽资源得到公平分配。

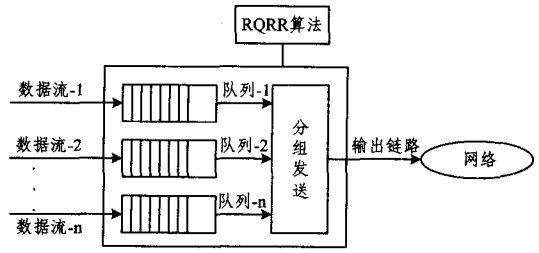

图 1 RQRR 算法调度示意图

2.1 相关概念

一个数据流是活动的(Active)，若这个数据流的分组正在出队列调度过程中被访问，或者这个数据流对应的队列非空。为了描述 RQRR 算法，将所有活动的数据流放在一个列表中，这个列表称为“活动数据流列表(ActiveFlowList)”。当一个数据流由活动的变成非活动的时，该数据流将从 ActiveFlowList 中移走。当一个数据流由非活动的变成活动的时，该数据流将加入到 ActiveFlowList 的尾部。

定义一个轮次(Round)是在某一时刻 T_1($T_1>0$)“活动数据流列表”中所包含的数据流被分组调度模块访问的过程。在本轮次执行过程中，新到达的或重新成为活动的数据流可以加入到“活动数据流列表”的尾部，但将从下一轮次开始被访问。例如轮次 1 的开始时间是 T_1，结束时间是 T_2($T_2>T_1$)。在 T_1 时刻，ActiveFlowList 中包含数据流－1、数据流－2、数据流－3 等 3 个数据流，那么在轮次 1，分组调度模块将访问上述的 3 个数据流。这时在 T_1、T_2 之间，数据流－4 变成活动的并加入到了 ActiveFlowList，但分组调度模块不会在轮次 1 访问数据流－4，因为数据流－4 在轮次 1 开始时刻 T_1 不在 ActiveFlowList 中。假设轮次 2 在 T_2 时刻开始，在轮次 2 中，分组调度模块将访问 T_2 时刻 ActiveFlowList 中的数据流，即访问数据流－1、数据流－2、数据流－3、数据流－4。因此，所说的第几轮次是相对于某个数据流来讲的，如从 T_2 开始的轮次，对于数据流－1、数据流－2、数据流－3 来说，是第 2 轮次，但对于数据流－4 来说，是第 1 轮次。

为了表示一个轮次中所需要访问的数据流的数目，引入一个计数器记录数据流的数目，这个计数器称为“访问数据流计数器(VisitFlowCount)”，表示在一个轮次开始时“活动数据流列表”中数据流的数量。每当分组调度模块访问一个数据流时，“访问数据流计数器”就减少 1，当“访问数据流计数器”最终等于 0 时，就说明这个轮次结束。

将一个数据流在一个轮次中所允许(Permission)发送的分组字节数称为这个数据流的 P 值，P 值就是前面提到的定额值(Quantum)。只是 P 值不是固定不变的，除了数据流的第 1 个轮次的 P 值为初值 0 外，其余轮次的 P 值都需要通过重新计算得到。计算方法是：所述数据流下一轮次的 P 值＝所述数据流本轮次 P 值＋本轮次其它数据流平均发送的分组字节数－所述数据流本轮次发送的字节数。在第 r(r 为大于 0 的自然数)个轮次中数据流－i($1\leqslant i\leqslant n$，i 为自然数)所允许发送的分组字节数记为 $P_i(r)$。

用一个计数器记录一个数据流在一个轮次中总共发送出去的分组字节数，这个计数器称为“字节数计数器”，每个数据流都有一个“字节数计数器”。在一个轮次开始前，“字节数计数器”的初值为 0，然后每发送一个分组，就将分组的字节数累加到对应数据流的“字节数计数器”。为了表述更直观，用

$Send$ 表示“字节数计数器”，用 $Send_i$ 表示数据流－i 的“字节数计数器”；用 $Send_i(r)$ 表示在第 r 轮次中从数据流－i 的队列中发送出去的分组字节数。

Average Count(AC)：对数据流－i 来说，AC 表示从第 r ($r\geqslant 2$)个轮次开始，除了数据流－i 之外的其余数据流在前一个轮次 $r-1$ 所发送的平均字节数，记为 $AC_i(r-1)$。如果第 $r-1$ 轮次总共有 n 个流，则 $AC_i(r-1)=[Send_1(r-1)+\cdots+Send_{i-1}(r-1)+Send_{i+1}(r-1)+\cdots+Send_n(r-1)]/(n-1)$，当不能整除时，取不小于商的整数，即商的整数部分＋1，可以用数学中向上取值操作实现。

于是，一个数据流 P 值的计算为：当 $r=1$ 时，数据流－i 的 P 值 $P_i(1)=0$；当 $r>1$ 时，$P_i(r)=P_i(r-1)+AC_i(r-1)-Send_i(r-1)$，($1\leqslant i\leqslant n$，$i$ 为自然数)，其意义是一个数据流在一个轮次中允许发送的数据量不仅与自己的 P 值有关，而且还依赖于其它数据流所发送的数据量。

2.2 实现

RQRR 算法包括初始化过程、入队列排队过程和出队列调度过程。在初始化过程中对分组调度模块进行初始化操作，包括初始化“活动数据流列表”和“访问数据流计数器”。入队列排队过程是指一个新的分组到达时，进入到与它所属数据流对应的队列中，等候发送。如果该分组所属的数据流不在“活动数据流列表”中，则将该分组所属的数据流加入到“活动数据流列表”的尾部；设置该数据流的 P 值为初值 0；设置该数据流的“字节数计数器”为初值 0。出队列调度过程是指从队列中选取分组，并通过输出链路发送出去，其流程如图 2 所示。

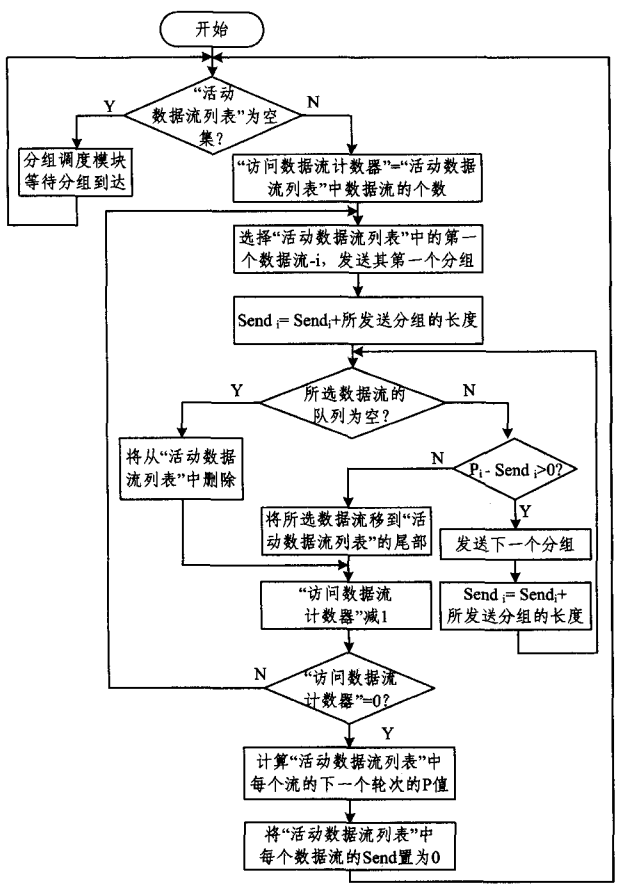

图 2 RQRR 算法的执行流程

RQRR 算法实现如下：

```
Initialization:
  ActiveFlowList=NULL;
  VisitFlowCount=0;
Enqueuing Module:on arrival of a packet
  i=QueueInWhichPacketArrives;
  if(ExistInActiveFlowList(i)==FALSE)then
    AddToActiveFlowList(i);
    Increment SizeOfActiveFlowList;
    Pi=0;
    Sendi=0;
  end if;
Dequeuing Module:
  while(TRUE)do
    if(ActiveFlowListIsEmpty==FALSE)then
      VisitFlowCount=SizeOfActiveFlowList;
    else
      waiting for arrival of a packet;
    end if;
    while(VisitFlowCount>0)do
      i=HeadOfActiveFlowList;
      RemoveHeadOfActiveFlowList;
      do
        TransmitPacketFromQueue(i);
        Sendi=Sendi+LengthOfTransmittedPackets;
      while(QueueIsEmpty==FALSE)&&(Pi-Sendi>0);
      if(QueueIsEmpty==FALSE)then
        AddQueueToActiveFlowList(i);
      else
        Decrement SizeOfActiveFlowList;
      end if;
      VisitFlowCount=VisitFlowCount-1;
    end while;
    for(i=1;i< SizeOfActiveFlowList+1;i=i+1)
      Pi=Pi+ACi-Sendi;
    for(i=1;i< SizeOfActiveFlowList+1;i=i+1)
      Sendi=0;
  end while;
```

2.3 举例

现有 3 个数据流：数据流－1 包含 4 个分组，数据流－2 包含 7 分组，数据流－3 包含 6 个分组，分组的排列及分组的长度(假设以字节为单位)如图 3 所示，左边的分组比右边的分组先期到达。

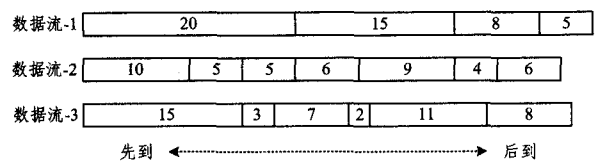

图 3 3 个数据流

当调用 RQRR 算法时，首先进行初始化，在此过程中，ActiveFlowList 的初值赋值为空列表；VisitFlowCount 的初值赋值为 0。

一个数据流的分组到达分组调度模块时，进入到该数据流的队列中排队，等候发送。如图 3 所示，数据流－1 的 4 个分组已经进入到数据流－1 的队列中，数据流－2 的 7 个分组已经进入到数据流－2 的队列中，数据流－3 的 6 个分组已经

进入到数据流－3的队列中。由于数据流－1、数据流－2、数据流－3等3个数据流的队列都非空，因此数据流－1、数据流－2、数据流－3都是“活动的”，3个数据流都加入到ActiveFlowList中，于是，ActiveFlowList＝{数据流－1，数据流－2，数据流－3}。数据流－1、数据流－2、数据流－3这3个数据流的 P 值赋初值为0，即当数据流－i 加入到ActiveFlowList中时，$P_i=0$，AC_i 的初值赋值为0，$Send_i$ 的初值赋值为0（$1\leqslant i\leqslant n, n=3$）。

执行RQRR算法的出队列调度过程，当ActiveFlowList非空时，分组调度模块开始对ActiveFlowList中数据流进行访问，从队列中选取分组并将分组从输出链路发送出去。RQRR算法对分组调度的4个轮次如图4所示，从左至右显示的内容是数据流标识、数据流本轮次的P值、发送的分组、数据流下一个轮次的P值计算。轮次4中没有数据流下一个轮次的P值计算，是由于ActiveFlowList已为空集，即此时已经没有数据流需要调度，因此，分组调度模块进入等待分组到达的状态。

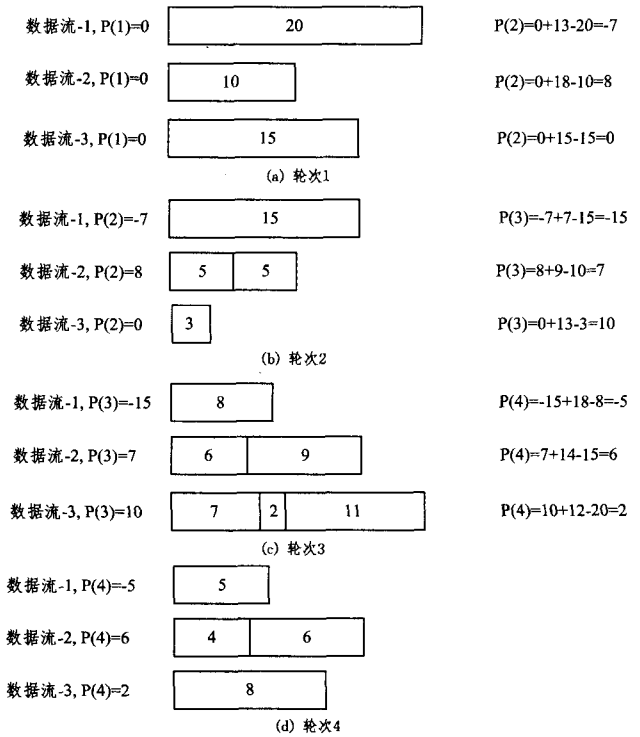

图4　RQRR算法调度的4个轮次

图4所示的分组调度过程，只是描述了在给定的图3所示数据流的情况下，执行RQRR算法分组调度开始的4个轮次。不过，RQRR算法对数据流数量、轮次的多少是没有限制的，一旦有数据流变成活动的，RQRR算法的分组调度模块还将继续往下执行分组调度。从图4也可以很容易看出，在一个轮次中获得很少服务（发送的字节数很少）的数据流会在下一个轮次中给予更多的服务机会。如数据流－3在轮次2中只发送了一个长度为3个字节的分组，而在轮次3就发送了3个分组，总长度达到20个字节。

3　性能分析

在这一节，将分析RQRR算法的运行复杂度以及RQRR算法的调度公平性。

3.1　复杂度

假设RQRR算法对 n 个数据流进行调度，定义RQRR算法的运行复杂度为一个分组进入队列排队，然后被调度出队列并从输出链路发送出去的时间复杂度。

定理1　RQRR算法的运行复杂度为O(1)。

证明：要证明定理1，只要能够证明一个分组进入队列排队的时间复杂度和一个分组被调度出队列的时间复杂度都为O(1)即可。

当一个新的分组到达时，执行RQRR算法的入队列排队过程，包含：确定这个新到达的分组应该进入哪一个队列，这个操作的时间复杂度为O(1)；如果这个分组所对应的数据流没有在ActiveFlowList中，就要将该数据流加入到ActiveFlowList的尾部，这个操作就是在一个链表的尾部增加一个数据项，它的时间复杂度也是O(1)。

现在分析调度一个分组出队列的时间复杂度。因为RQRR算法在对一个数据流进行操作时，都会从这个数据流的队列中选择至少一个分组发送出去，所以调度一个分组出队列的时间复杂度等于或小于执行RQRR算法出队列调度过程所有操作的时间复杂度。这些操作包括确定下一个要访问的数据流，从ActiveFlowList的首部移出一个数据流，有可能再把数据流加入到ActiveFlowList的尾部，这些在链表上进行的操作都可以在O(1)的时间内完成。另外，还包括更新变量（VisitFlowCount、$Send_i$、P_i）的值等操作，这些操作都可以在常数时间内完成。所以，调度一个分组出队列的时间复杂度为O(1)。证毕。

3.2　公平性

在公平性分析中，采用最早在文献[5]中提出的公平性评价方法，用 $Send_i(t_1,t_2)$ 表示在时间 t_1、t_2 之间从数据流 i 发送出去的字节数。于是，在时间间隔 (t_1,t_2)，定义公平性度量 $FM(t_1,t_2)$ 为任意两个活动数据流 i、j 所发送字节数之差中的最大值，即

$FM(t_1,t_2)=\max(|Send_i(t_1,t_2)-Send_j(t_1,t_2)|)$，给定时间间隔 (t_1,t_2)

在这里只考虑网络中的活动数据流，是因为RQRR算法不会对非活动数据流进行任何操作，所以不需要考虑与非活动数据流之间的比较。

定义 FM 是所有可能时间间隔 (t_1,t_2) 中 $FM(t_1,t_2)$ 的最大值，FM 越接近0，说明公平性越好。但是，对任何以分组为传输单位的算法就难以满足这样的条件，只能要求它们的 FM 是一个较小的常数。因此，如果一个调度算法的 FM 是一个常数，特别地，$FM(t_1,t_2)$ 不依赖于时间间隔的大小，那么称该调度算法是接近公平的。

定义Max为分组的最大长度，即最大数据分组的字节数。

性质1　在 $AC_i(1\leqslant i\leqslant n)$ 不进行取整运算的情况下，对任意轮次 $r(r\geqslant 1)$，n 个数据流的 P 值之和等于0。

证明：对于 $r=1$ 的情况，因为 $P_i(1)=0$，所以 n 个数据流的 P 之和等于0。下面分析 $r\geqslant 2$ 的情形，因为 $P_i(r)=P_i(r-1)+AC_i(r-1)-Send_i(r-1)$，所以 n 个数据流的 P 值之和为：

$$\sum_{i=1}^{n}P_i(r)=\sum_{i=1}^{n}[P_i(r-1)+AC_i(r-1)-Send_i(r-1)]$$
$$=\sum_{i=1}^{n}P_i(r-1)+\sum_{i=1}^{n}AC_i(r-1)-\sum_{i=1}^{n}Send_i(r-1)$$

因为$\sum_{i=1}^{n}AC_i(r-1)=\frac{1}{n-1}\times(n-1)\sum_{i=1}^{n}Send_i(r-1)=\sum_{i=1}^{n}Send_i(r-1)$，所以，

$$\sum_{i=1}^{n}P_i(r)=\sum_{i=1}^{n}P_i(r-1)+0=\sum_{i=1}^{n}P_i(r-1)$$

因为 $P_i(1)=0$，所以$\sum_{i=1}^{n}P_i(r)=0$。证毕。

引理 1 对任意数据流 $i(1\leqslant i\leqslant n)$和轮次 $r(r\geqslant 1)$，都有 $P_i(r)<2\text{Max}$。

证明：当 $r=1$ 时，$P_i(1)=0$，结论成立。

当 $r\geqslant 2$ 时，$P_i(r)=P_i(r-1)+AC_i(r-1)-Send_i(r-1)$。因为 $Send_i(r-1)\geqslant P_i(r-1)$，所以 $P_i(r)\leqslant AC_i(r-1)$。

由性质 1 可知，数据流的 P 值如果不全为 0，就不可能全为正数，而且正 P 值之和与负 P 值之和在数值上是相等的。当数据流的 P 值为负时，只允许发送 1 个分组，其最大长度为 Max。不失一般性，假设在轮次 $r-1$ 数据流 1 的 P 值为负值，则数据流 1 发送的最大分组长度为 Max；又假设数据流 2 是 P 值最大的数据流，且 P 值大于 Max 达到 2Max−1。数据流 1 和 2 能够发送的分组长度之和最大为 Max+2Max−2+Max=4Max−2，平均值为 2Max−1，小于 2Max。如果要求 $AC_i(r-1)\geqslant 2\text{Max}$，则其它数据流 $j(j\neq i)$的 $P_j(r-1)$的平均值要大于 Max，也就要求 $AC_j(r-2)$的平均值大于 Max，依此类推，就要求 $AC_j(1)$的平均值大于 Max，但 $AC_j(1)$的平均值最大只能为 Max。于是 $AC_i(r-1)<2\text{Max}$，所以 $P_i(r)<2\text{Max}$。证毕。

定理 2 设从轮次 $k(k\geqslant 1)$开始的 m 个连续的轮次中，数据流 $i(1\leqslant i\leqslant n)$是活动的，且数据流 i 在连续的 m 个轮次所发送出去的数据分组长度总量记为 Sum_i。对任意的轮次 $r(k\leqslant r\leqslant k+m-1)$，如果 $P_i(r)\geqslant 0$，则有

$$-2\text{Max}+\sum_{r=k}^{k+m-1}AC_i(r)<Sum_i<\sum_{r=k}^{k+m-1}AC_i(r)+2\text{Max}$$

证明：因为当 $r\geqslant 2$ 时，$P_i(r)=P_i(r-1)+AC_i(r-1)-Send_i(r-1)$，所以 $Send_i(r-1)=P_i(r-1)-P_i(r)+AC_i(r-1)$，于是

$$\begin{aligned}Sum_i&=\sum_{r=k}^{k+m-1}Send_i(r)\\&=\sum_{r=k}^{k+m-1}[P_i(r)-P_i(r+1)+AC_i(r)]\\&=\sum_{r=k}^{k+m-1}[P_i(r)-P_i(r+1)]+\sum_{r=k}^{k+m-1}AC_i(r)\\&=P_i(k)-P_i(k+m)+\sum_{r=k}^{k+m-1}AC_i(r)\end{aligned}$$

因为 $P_i(r)<0$ 与 $P_i(r)=0$ 都只允许数据流 i 发送一个分组，所以当 $P_i(r)<0$ 时，令 $P_i(r)=0$，于是有 $P_i(r)\geqslant 0$。又由引理 1 有 $P_i(r)<2\text{Max}$，所以 $-2\text{Max}<P_i(k)-P_i(k+m)<2\text{Max}$，故定理 2 得证。

下面的定理 3 说明 RQRR 算法的公平性度量有一个与时间间隔大小无关的常数上界值。

定理 3 如果在 RQRR 算法运行过程中，当 $P_i(r)<0$ 时，令 $P_i(r)=0$，$1\leqslant i\leqslant n$，则对任意时间间隔(t_1,t_2)，有 $FM(t_1,t_2)<7\text{Max}-1$。

证明：现考虑两个在时间间隔(t_1,t_2)活动的数据流 i 和 j，RQRR 由于是轮询算法，因此会轮流访问 ActiveFlowList 中的数据流，易知连续两次访问数据流 i 之间会对数据流 j 访问一次。设 m_i 是在时间间隔(t_1,t_2)访问数据流 i 的次数，m_j 是在时间间隔(t_1,t_2)访问数据流 j 的次数，于是有 $|m_i-m_j|\leqslant 1$。

不失一般性，假设从轮次 k 开始，数据流 i 在数据流 j 之前被访问，这样 $m_j=m_i$ 或 $m_j=m_i-1$。由定理 2 有

$$Send_i(t_1,t_2)<\sum_{r=k}^{k+m_i-1}AC_i(r)+2\text{Max} \tag{1}$$

$$Send_j(t_1,t_2)>-2\text{Max}+\sum_{r=k}^{k+m_j-1}AC_j(r) \tag{2}$$

所以，

$$\begin{aligned}&Send_i(t_1,t_2)-Send_j(t_1,t_2)\\&<\sum_{r=k}^{k+m_i-1}AC_i(r)+2\text{Max}-[-2\text{Max}+\sum_{r=k}^{k+m_j-1}AC_j(r)]\\&=4\text{Max}+\sum_{r=k}^{k+m_i-1}AC_i(r)-\sum_{r=k}^{k+m_j-1}AC_j(r)\end{aligned} \tag{3}$$

当 $m_j=m_i$ 时，

$$\begin{aligned}&\sum_{r=k}^{k+m_i-1}AC_i(r)-\sum_{r=k}^{k+m_j-1}AC_j(r)=\sum_{r=k}^{k+m_i-1}AC_i(r)-\sum_{r=k}^{k+m_i-1}AC_j(r)\\&=\sum_{r=k}^{k+m_i-1}[AC_i(r)-AC_j(r)]\\&=\frac{1}{n-1}\sum_{r=k}^{k+m_i-1}[Send_j(r)-Send_i(r)]\\&=\frac{1}{n-1}[\sum_{r=k}^{k+m_i-1}Send_j(r)-\sum_{r=k}^{k+m_i-1}Send_i(r)]\\&=\frac{1}{n-1}[Send_j(t_1,t_2)-Send_i(t_1,t_2)]\end{aligned}$$

所以，由式(3)有

$$Send_i(t_1,t_2)-Send_j(t_1,t_2)<4\text{Max}+\frac{1}{n-1}[Send_j(t_1,t_2)-Send_i(t_1,t_2)]$$

即

$$(1+\frac{1}{n-1})[Send_i(t_1,t_2)-Send_j(t_1,t_2)]<4\text{Max}$$

得到

$$Send_i(t_1,t_2)-Send_j(t_1,t_2)<\frac{n-1}{n}\times 4\text{Max}<4\text{Max} \tag{4}$$

当 $m_j=m_i-1$ 时，

$$\begin{aligned}&\sum_{r=k}^{k+m_i-1}AC_i(r)-\sum_{r=k}^{k+m_j-1}AC_j(r)=\sum_{r=k}^{k+m_i-1}AC_i(r)-\sum_{r=k}^{k+m_i-2}AC_j(r)\\&=\sum_{r=k}^{k+m_i-1}AC_i(r)-\sum_{r=k}^{k+m_i-1}AC_j(r)+AC_j(k+m_i-1)\\&=\sum_{r=k}^{k+m_i-1}[AC_i(r)-AC_j(r)]+AC_j(k+m_i-1)\\&=\frac{1}{n-1}\sum_{r=k}^{k+m_i-1}[Send_j(r)-Send_i(r)]+AC_j(k+m_i-1)\\&=\frac{1}{n-1}[\sum_{r=k}^{k+m_i-1}Send_j(r)-\sum_{r=k}^{k+m_i-1}Send_i(r)]+AC_j(k+m_i-1)\\&=\frac{1}{n-1}[\sum_{r=k}^{k+m_j-1}Send_j(r)+Send_j(k+m_j)-\sum_{r=k}^{k+m_i-1}Send_i(r)]+AC_j(k+m_i-1)\\&=\frac{1}{n-1}[Send_j(t_1,t_2)-Send_i(t_1,t_2)]+\frac{1}{n-1}Send_j(k+m_j)+AC_j(k+m_j)\\&=\frac{1}{n-1}[Send_j(t_1,t_2)-Send_i(t_1,t_2)]+\frac{1}{n-1}\sum_{u=1}^{n}Send_u(k+m_j)\end{aligned}$$

所以，由式(3)有

$$Send_i(t_1,t_2)-Send_j(t_1,t_2)<4\text{Max}+\frac{1}{n-1}[Send_j(t_1,t_2)-Send_i(t_1,t_2)]+\frac{1}{n-1}\sum_{u=1}^{n}Send_u(k+m_j)$$

即

$$(1+\frac{1}{n-1})[Send_j(t_1,t_2)-Send_i(t_1,t_2)]<4\text{Max}+\frac{1}{n-1}\sum_{u=1}^{n}Send_u(k+m_j)$$

得到

$$Send_j(t_1,t_2)-Send_i(t_1,t_2)<\frac{n-1}{n}\times 4\text{Max}+\frac{1}{n}\sum_{u=1}^{n}Send_u(k+m_j)$$

因为 $Send_u(k+m_j)\leqslant P_u(k+m_j)-1+\text{Max}<2\text{Max}-1+\text{Max}=3\text{Max}-1$，所以，

$$Send_j(t_1,t_2)-Send_i(t_1,t_2)<\frac{n-1}{n}\times 4\text{Max}+\frac{1}{n}\times n\times(3\text{Max}-1)<7\text{Max}-1 \quad (5)$$

结合式(4)、式(5)可得 $FM(t_1,t_2)<7\text{Max}-1$。证毕。

由定理 3 的证明可以得到下列推论：

推论 1 对活动数据流 i 和 $j(1\leqslant i,j\leqslant n)$，如果在连续相同数量的轮次中 P_i 和 P_j 都大于等于 0，那么，数据流 i 和 j 所发送出去的数据分组长度总量之差小于 4Max。

3.3 对比分析

FQ(Fair Queuing)[1]提出了一种理想的公平队列算法，称为 bit-by-bit round-robin(BR)，它以轮询的方式访问每个数据流队列，每个数据流每次发送一个比特(bit)。从分组的角度来看它的公平性，其 FM 不会超过最大分组的长度，即 $FM\leqslant\text{Max}$。但真正实现这样一个系统是不现实的，只能是近似地模拟 BR，模拟的时间复杂度是 $O(\log(N))$，这里 N 是指数据流的个数。因此，BR 虽然保证了公平性，但不易在高速网络中得以实现。SCFQ(Self-Clocked Fair Queuing)[5]的时间复杂度也是 $O(\log(N))$，$FM\leqslant 2\text{Max}$。

DRR 算法在文献[7]中已得到证明，当所有的定额值都大于或等于 Max 时，每次访问一个队列至少可以发送一个分组，得到 DRR 算法的运行复杂度为 O(1)。假设 Q 是所有定额值中最小的，则 DRR 算法有 $FM\leqslant 2\text{Max}+Q$。为了满足复杂度 O(1)的要求，需要 $Q\geqslant\text{Max}$，为此取 $Q=\text{Max}$，则得到 $FM\leqslant 3\text{Max}$，所以 DRR 算法具有比较好的公平性，不过 Q 的选取存在不确定性。PNDRR 算法[12]将低带宽低延迟业务汇聚到一个队列中优先处理，改变了嵌套式 DRR 算法进入服务链表的规则，使得优先队列中的分组尽早获得服务。但是，算法引入的虚令牌数 Q_c 和漏桶的 σ 参数必须满足相应的条件，PNDRR 算法才能满足公平性的要求。PNDRR 算法的时间复杂度与 DRR 算法一样，在保证 $Q\geqslant\text{Max}$ 的条件下为 O(1)。

文献[13]中虽然加入整形器使信息流输出较为平滑，但是与没有整形器的情况相比，信息流的延迟差异相对较大，影响了 DRR 算法原有的对带宽分配的公平性优点。文献[15]提出的是一种用于分组交换路由的分层调度算法，其将基于时间戳和轮询的调度算法进行结合，其中承载组内的调度采用的是 Smoothed DRR 算法，而域间调度采用的则是最长队列优先调度算法，算法的公平性较好，但复杂度仍较高。文献[16]提出的是在考虑无线通信条件下的 DRR 算法，文中主要考察了算法对系统吞吐量的影响。文献[17]利用 DRR 将不同数据类型的传输要求统一到对定额值(Quanta)的计算，根据文中定义的公平性度量 Utilization-based Fairness Measure (UFM)，DRR 可以帮助所有的数据流获得 90%左右的相对公平性。

表 1 中列出了一些分组调度算法的公平性和复杂度。现阶段能支持变长分组调度的轮询算法中，许多算法都是基于 DRR 的改进算法或者是将 DRR 用于某一特定的应用领域。相比之下，本文提出的 RQRR 算法是一种能支持变长分组调度的新算法，且具有确定的公平性和 O(1)的实现复杂度。

表 1 公平性和复杂度对比

调度算法	公平性	复杂度
Round Robin	∞	O(1)
First-Come-First-Serve	∞	O(1)
Fair Queuing(BR)	Max	O(log(N))
Self-Clocked Fair Queuing(SCFQ)	2Max	O(log(N))
Deficit Round Robin(DRR)	2Max+Q	O(1)
Prioritized Nested DRR(PNDRR)	有条件支持	O(1)
Resilient Quantum Round Robin(RQRR)	7Max−1	O(1)

结束语 在高速网络中，适用于变长分组的轮询调度算法受到重视。本文提出了一种新的适用于变长分组的调度算法 RQRR(Resilient Quantum Round Robin)，与 DRR 不同，一个数据流在一个轮次中所允许发送的字节数，即一个数据流的定额值，不是对所有的轮次都是固定不变的，而是依赖于前一个轮次中各个数据流的发送情况进行计算。RQRR 的运行复杂度为 O(1)，因此易于在数据流较多的高速网络中实现。RQRR 的公平性度量具有常数上界值 7Max−1，其中 Max 为最大分组长度，可以保证数据流之间具有较好的调度公平性。相对于 DRR 算法，RQRR 不要求在发送分组前知道分组的长度，而且保证一个数据流在一个轮次中至少有一个分组被发送。本文主要研究了 RQRR 算法的公平性和实现复杂度，下一步工作将对 RQRR 算法的时延特性进行分析。

参 考 文 献

[1] Demers A, Keshav S, Shenker S. Analysis and simulation of a fair queueing algorithm[J]. Proc. SIGCOMM '89, 1989, 19(4): 1-12

[2] Keshav S. On the efficient implementation of fair queuing[J]. Journal of Internetworking Research and Experience, 1991, 2(3): 1-20

[3] Bennett J C R, Zhang H. WF2Q: Worst-case fair weighted fair queueing[C]// INFOCOM '96. San Fran-ciso, California, Mar. 1996: 120-128

[4] Zhang L. Virtual clock: A new traffic control algorithm for packet switching networks[C]// Proceedings of ACM SIGCOMM. Philadelphia, September 1990: 19-29

[5] Golestani S. A self-clocked fair queueing scheme for broadband applications[C]//INFOCOM'94. June 1994: 636-646

[6] Parekh A. A generalized processor sharing approach to flow control in integrated services network[D]. Dept. Elect. Eng. and Comput. Sci., M. I. T., Feb. 1992

[7] Shreedhar M, Varghese G. Efficient Fair Queuing Using Deficit Round Robin [J]. IEEE/ACM Transaction on Networking (S1063-6692), 1996, 4(3): 375-385

[8] Sivakumar G, Ramprasad A V. Analysis of FiWi networks to improve TCP Performance[C]// 2012 International Conference on Computing, Communication and Applications (ICCCA). 2012:1-6

[9] Ayaz S, Hoffmann F, German R, et al. Analysis of deficit round robin scheduling for future aeronautical data link[C]// 2011 IEEE 22nd International Symposium on Personal Indoor and Mobile Radio Communications(PIMRC). 2011:1809-1814

[10] Sleem M Y, ElBadawy H M, Abo-El-Seoud M S. Two layer channel aware scheduling for QoS support in IEEE 802.16/WiMAX networks[C]// 2011 Eighth International Conference on Wireless and Optical Communications Networks(WOCN). 2011:1-5

[11] Mansy A, Ammar M, Zegura E. Deficit Round-Robin Based Message Ferry Routing[C]// Global Telecommunications Conference(GLOBECOM 2011). 2011:1-5

[12] 简贵宵,葛宁,冯重熙. 具有优先服务机制的嵌套式 DRR 算法[J]. 电子与信息学报,2005,27(1):123-126

[13] 高斐,张原,杨百战. 差额轮循的平滑输出算法研究[J]. 西北工业大学学报,2011,29(1):49-53

[14] Kumar D, Priyameenal V. Adaptive packet scheduling algorithm for real-time services in Wi-MAX networks[C]// 2011 International Conference on Recent Trends in Information Technology (ICRTIT). 2011:342-347

[15] 张博,汪斌强,王珊珊,等. 基于 Crossbar 的可重构网络输入排队分域调度研究[J]. 通信学报,2012,33(9):105-115

[16] Yang Shu-qing, Peng Jin-ye. QoS Multi-users Packet Scheduling Algorithm in IEEE 802.16e System[C]// 2011 International Conference on Computer and Automation Engineering(ICCAE 2011). IPCSIT vol. 44, 2012:112-116

[17] Yu Hua, Liu Xue. Scheduling Heterogeneous Flows with Delay-Aware Deduplication for Avionics Applications[J]. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(9):1790-1802

(上接第 48 页)

关值与相关均值的比值来进行,且将判据阈值设为 6[2],捕获结果如图 8 所示。以捕获概率为 90%以上作为标准,传统相干积分法能捕获到信噪比低至-40dB 的信号,而全比特算法能捕获到信噪比低至-50dB 的信号,本文算法介于二者之间,能捕获到信噪比低至-47dB 的信号。

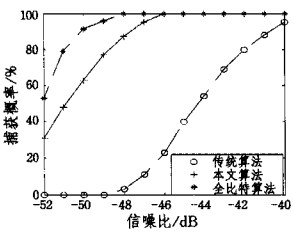

图 8 3 种算法在不同信噪比下的捕获效果比较

随着信噪比的降低,非相干积分所带来的平方损耗影响也将增大,到了一定程度有可能会出现峰值不够明显,出现多个峰值甚至没有峰值出现等情况,造成捕获失败,因此,随着噪声的增大,接收机捕获的成功率将会有所下降,在本文的仿真条件下,本文算法最低能捕获到信噪比为-47dB 的信号。

另外,接收机的捕获性能除与算法的性能有关之外,也与捕获采用的数据长度有关。本文采用 140ms 的数据进行捕获,可用于室内条件下微弱信号(信噪比为-44dB 左右[9])的捕获。若需捕获更为微弱的信号,则需进一步增加积分长度或采用更为灵敏的捕获算法。

由此可以看出,本文算法相比全比特算法计算量大大减少,又能取得良好的捕获效果,是一种综合性能良好的算法。

结束语 本文介绍了一种基于位跳变检测的微弱信号快速捕获算法,它通过估计位跳变边缘位置避开跳变位对相干积分的影响,并延长相干积分时间。仿真实验效果表明,该算法采用 140ms 的数据可以捕获信噪比低至-47dB 的信号,有效地提高了 GPS 软件接收机的捕获灵敏度,同时也可以消除导航数据位跳变引起的误捕获,算法的运算效率与全比特算法和位跳变检测算法相比大幅提高。

参 考 文 献

[1] 林庆恩,茅旭初. 一种用于微弱 GPS 信号处理的快速捕获方法[J]. 计算机仿真,2010,27(10):330-334

[2] 黎山,易清明,陈庆,等. 适用于 GPS 软件接收机的弱信号捕获方法[J]. 计算机应用,2012,32(3):816-822

[3] Ba Xiao-hui, et al. A novel algorithm based on FFT for ultra high-sensitivity GPS Tracking[C]// IONGNSS2009. Savannah, 2009:339-345

[4] Mohammadreza Z, et al. Enhanced GNSS indoor signal detectability using polarization diversity[C]// IONGNSS 2009. Savannah, 2009:395-205

[5] Cai Y, et al. Using network RTK corrections and low-cost receiver for precise mass market positioning and navigation applications[C]// IEEE Intelligent Vehicles Symposium. Baden-Baden, 2011:345-349

[6] Hu Cong-wei, el al. Assisted GPS positioning under weak signal environments[C]// ION GNSS 2009. Savannah, 2009:3102-3109

[7] Wu Ling-juan, Lu Wei-jun, Yu Dun-shan. Research of weak signal acquisition algorithms for high sensitivity GPS receivers[C]// PrimeAsia 2009. Shanghai, 2009:173-176

[8] Psiaki M L. Block Acquisition of weak GPS signals in a Software Receiver[C]// ION GPS 2001. Salt Lake City, 2001

[9] 马若飞. GPS 弱信号捕获算法研究及其在软件接收机上的实现[D]. 哈尔滨:哈尔滨工业大学,2010:17-40

[10] Kazemi P L, O'Driscoll C. Comparison of assisted and stand-alone methods for increasing coherent integration time for weak GPS signal tracking[C]// ION GNSS 2008. Savannah, 2008:1043-1053

[11] 何秋生,程亚奇. 基于位跳变检测提高 GPS 信号处理增益方法研究[J]. 微电子学与计算机,2010,27(2):99-102

[12] Wu Ling-juan, Cui Ying-ying, Yu Dun-shan. Signal Acquisition and Tracking for Software GPS[C]// PrimeAsia 2009. Shanghai, 2009:384-387

[13] 裘勋,卢艳娥,庞春雷. 改进的 GPS 弱信号差分捕获方法研究[J]. 现代防御技术,2011,39(6):126-130